WHAT IS CLAIMED IS:

1 1. A computer-implemented method for using a processor as
2   a virtual device, said method comprising:

3   signaling, from a first processor, a second processor
4   from a plurality processors in a computer system,
5   wherein the plurality of processors share a common
6   memory and wherein at least two of the processors are
7   dislike;

8   storing data corresponding to the request in a local
9   memory corresponding to the second processor; and

10   processing the data by the second processor using
11   software code stored in the second processor's local
12   memory.

1 2. The method as described in claim 1 further comprising:

2   writing, from a software program executing on the
3   first processor, the data into an input buffer stored
4   in the common memory, wherein the storing further
5   includes sending the data from the input buffer into
6   the second processor's local memory; and

7   loading the software code from the common memory to
8   the second processor's local memory prior to the
9   processing.

1 3. The method as described in claim 2 wherein the sending
2   of the data into the second processor's local memory
3   and the loading of the software code are both
4   performed using DMA operations.

1   4.   The method as described in claim 3 wherein the DMA
2   operations are performed using one of a plurality of
3   DMA controllers, one of which being assigned to the
4   first processor and a second of which being assigned
5   to the second processor.

1   5.   The method as described in claim 1 further comprising:

2   generating result data, stored in the second
3   processor's local memory, in response to the
4   processing.

1   6.   The method as described in claim 5 further comprising:

2   writing the result data to an output buffer stored in
3   the common memory.

1   7.   The method as described in claim 5 further comprising:

2   writing the result data to an input buffer that
3   corresponds with another device.

1   8.   The method as described in claim 7 further comprising:

2   signaling, from the second processor, a third
3   processor from the plurality processors, wherein the
4   third processor includes a local memory;

5   storing the result data in the third processor's local
6   memory; and

7   processing the result data by the third processor
8   using software code stored in the third processor's
9   local memory.

1   9.   The method as described in claim 5 further comprising:

2   writing the result data to a physical device.

1   10.   The method as described in claim 1 wherein the

2   software code stored in the second processor's local

3   memory includes a plurality of code routines, the

4   method further comprising:

5   identifying one of the code routines based upon the

6   request; and

7   executing the identified code routine.

1   11.   The method as described in claim 1 further comprising:

2   initializing a task queue for one or more device

3   functions to be performed by one or more of the

4   plurality of processors, wherein the signaling further

5   includes:

6   writing the request to the task queue;

7   determining, by the second processor, that the request

8   is in the task queue; and

9   reading the request from the task queue in response to

10   the determination.

1   12.   The method as described in claim 1 further comprising:

2   generating result data, stored in the second

3   processor's local memory, in response to the

4   processing.

1   13.   The method as described in claim 1 further comprising:

2   prior to the signaling:

3   loading the data to a first location in the
4   common memory;

5   loading the software code adapted to perform the
6   processing to a second location in the common
7   memory;

8   writing an instruction block to a third location
9   in the common memory, wherein the instruction
10   block includes the first and second locations;

11   wherein the signaling includes writing the third
12   location to a mailbox corresponding to the second
13   processor.

1   14.   The method as described in claim 13 further
2   comprising:

3   reading, by the second processor, the third location
4   from the second processor's mailbox;

5   reading, by the second processor, the first and second
6   locations from the instruction block stored at the
7   third location; and

8   reading, by the second processor, the data from the
9   first location in the common memory prior to storing
10   the data in the second processor's local memory.

1   15.   The method as described in claim 13 further
2   comprising:

3   reading, by the second processor, the software code
4   stored at the second location in the common memory;
5   and

6   storing the software code in the second processor's

7   local memory.

1   16.   The method as described in claim 15 wherein the

2   reading and storing of the software code are performed

3   in response to determining that the software code

4   stored at the second location in the common memory is

5   not already stored in the second processor's local

6   memory.

1   17.   An information handling system comprising:

2   a plurality of heterogeneous processors;

3   a common memory shared by the plurality of

4   heterogeneous processors;

5   a first processor selected from the plurality of

6   processors that sends a request to a second processor,

7   the second processor also being selected from the

8   plurality of processors;

9   a local memory corresponding to the second processor;

10   a DMA controller associated with the second processor,

11   the DMA controller adapted to transfer data between

12   the common memory and the second processor's local

13   memory; and

14   a virtual device tool for operating the second

15   processor as a virtual device, the virtual device tool

16   including software effective to:

17   signal, from the first processor, the second

18   processor;

19   store data corresponding to the request in the
20   second processor's local; and

21   process the data by the second processor using
22   software code stored in the second processor's
23   local memory.

1   18.   The information handling system as described in claim
2   17 further comprising software effective to:

3   write, from a software program executing on the first
4   processor, the data into an input buffer stored in the
5   common memory, wherein the storing further includes
6   sending the data from the input buffer into the second
7   processor's local memory; and

8   load the software code from the common memory to the
9   second processor's local memory prior to the
10   processing.

1   19.   The information handling system as described in claim
2   18 wherein the sending of the data into the second
3   processor's local memory and the loading of the
4   software code are both performed using DMA operations
5   with the DMA controller.

1   20.   The information handling system as described in claim
2   17 further comprising software effective to:

3   generate result data, stored in the second processor's
4   local memory, in response to processing the data using
5   the software code.

1   21.   The information handling system as described in claim

2   20 further comprising software effective to:

3   write the result data to an output buffer stored in

4   the common memory.

1   22.   The information handling system as described in claim

2   20 further comprising software effective to:

3   write the result data to an input buffer that

4   corresponds with another device.

1   23.   The information handling system as described in claim

2   22 further comprising software effective to:

3   signal, from the second processor, a third processor

4   from the plurality processors, wherein the third

5   processor includes a local memory;

6   store the result data in the third processor's local

7   memory; and

8   process the result data by the third processor using

9   software code stored in the third processor's local

10   memory.

1   24.   The information handling system as described in claim

2   20 further comprising software effective to:

3   write the result data to a physical device.

1   25.   The information handling system as described in claim

2   17 wherein the software code stored in the second

3   processor's local memory includes a plurality of code

4   routines, the information handling system further

5   comprising software effective to:

6   identify one of the code routines based upon the

7   request; and

8   execute the identified code routine.

1   26.   The information handling system as described in claim

2   17 further comprising software effective to:

3   initialize a task queue for one or more device

4   functions to be performed by one or more of the

5   plurality of processors, wherein the signal of the

6   second processor further includes software effective

7   to:

8   write the request to the task queue;

9   determine, by the second processor, that the request

10   is in the task queue; and

11   read the request from the task queue in response to

12   the determination.

1   27.   The information handling system as described in claim

2   17 further comprising software effective to:

3   generate result data, stored in the second processor's

4   local memory, in response to the processing of the

5   data using the software code.

1   28.   The information handling system as described in claim

2   17 further comprising, prior to the signal of the

3   second processor, software effective to:

4   loading the data to a first location in the

5   common memory;

6   loading the software code adapted to perform the

7   processing to a second location in the common

8   memory;

9   writing an instruction block to a third location

10   in the common memory, wherein the instruction

11   block includes the first and second locations;

12   wherein the signal of the second processor includes

13   software code effective to write the third location to

14   a mailbox corresponding to the second processor.

1 29. The information handling system as described in claim

2   28 further comprising software to:

3   read, by the second processor, the third location from

4   the second processor's mailbox;

5   read, by the second processor, the first and second

6   locations from the instruction block stored at the

7   third location; and

8   read, by the second processor, the data from the first

9   location in the common memory prior to storing the

10   data in the second processor's local memory.

1 30. The information handling system as described in claim

2   28 further comprising software effective to:

3   read, by the second processor, the software code

4   stored at the second location in the common memory;

5   and

6   store the software code in the second processor's

7   local memory.

1   31. The information handling system as described in claim

2   30 wherein the reading and storing of the software

3   code are performed in response to determining that the

4   software code stored at the second location in the

5   common memory is not already stored in the second

6   processor's local memory.

1   32. A computer program product stored on a computer

2   operable media for using a processor as a virtual

3   device, said computer program product comprising:

4   means for signaling, from a first processor, a second

5   processor from a plurality processors in a computer

6   system, wherein the plurality of processors share a

7   common memory and wherein at least two of the

8   processors are dislike;

9   means for storing data corresponding to the request in

10   a local memory corresponding to the second processor;

11   and

12   means for processing the data by the second processor

13   using software code stored in the second processor's

14   local memory.

1   33. The computer program product as described in claim 32

2   further comprising:

3   means for writing, from a software program executing

4   on the first processor, the data into an input buffer

5   stored in the common memory, wherein the means for

6   storing further includes means for sending the data

7   from the input buffer into the second processor's

8   local memory; and

9   means for loading the software code from the common

10   memory to the second processor's local memory prior to

11   the processing.

1 34. The computer program product as described in claim 33

2   wherein the means for sending the data into the second

3   processor's local memory and the means for loading the

4   software code are both performed using DMA operations.

1 35. The computer program product as described in claim 34

2   wherein the DMA operations are performed using one of

3   a plurality of DMA controllers, one of which being

4   assigned to the first processor and a second of which

5   being assigned to the second processor.

1 36. The computer program product as described in claim 32

2   further comprising:

3   means for generating result data, stored in the second

4   processor's local memory, in response to the

5   processing.

1 37. The computer program product as described in claim 36

2   further comprising:

3   means for writing the result data to an output buffer

4   stored in the common memory.

1 38. The computer program product as described in claim 36

2   further comprising:

3   means for writing the result data to an input buffer

4   that corresponds with another device.

1 39. The computer program product as described in claim 38

2   further comprising:

3   means for signaling, from the second processor, a

4   third processor from the plurality processors, wherein

5   the third processor includes a local memory;

6   means for storing the result data in the third

7   processor's local memory; and

8   means for processing the result data by the third

9   processor using software code stored in the third

10   processor's local memory.

1 40. The computer program product as described in claim 36

2   further comprising:

3   means for writing the result data to a physical

4   device.

1 41. The computer program product as described in claim 32

2   wherein the software code stored in the second

3   processor's local memory includes a plurality of code

4   routines, the computer program product further

5   comprising:

6   means for identifying one of the code routines based

7   upon the request; and

8   means for executing the identified code routine.

1 42. The computer program product as described in claim 32

2   further comprising:

3   means for initializing a task queue for one or more

4   device functions to be performed by one or more of the

5   plurality of processors, wherein the signaling further

6   includes:

7   means for writing the request to the task queue;

8   means for determining, by the second processor, that

9   the request is in the task queue; and

10   means for reading the request from the task queue in

11   response to the determination.

1   43.   The computer program product as described in claim 32

2   further comprising:

3   means for generating result data, stored in the second

4   processor's local memory, in response to the

5   processing.

1   44.   The computer program product as described in claim 32

2   further comprising:

3   prior to the means for signaling:

4   means for loading the data to a first location in

5   the common memory;

6   means for loading the software code adapted to

7   perform the processing to a second location in

8   the common memory; and

9   means for writing an instruction block to a third

10   location in the common memory, wherein the

11   instruction block includes the first and second

12   locations;

13   wherein the means for signaling includes means for

14   writing the third location to a mailbox corresponding

15   to the second processor.

1   45.   The computer program product as described in claim 44

2   further comprising:

3   means for reading, by the second processor, the third

4   location from the second processor's mailbox;

5   means for reading, by the second processor, the first

6   and second locations from the instruction block stored

7   at the third location; and

8   means for reading, by the second processor, the data

9   from the first location in the common memory prior to

10   storing the data in the second processor's local

11   memory.

1   46.   The computer program product as described in claim 44

2   further comprising:

3   means for reading, by the second processor, the

4   software code stored at the second location in the

5   common memory; and

6   means for storing the software code in the second

7   processor's local memory.

1   47.   The computer program product as described in claim 46

2   wherein the means for reading and means for storing

3   the software code are performed in response to

4   determining that the software code stored at the

5   second location in the common memory is not already

6   stored in the second processor's local memory.